

1

LAMMP, Now with an Extra M

How things have changed in the last decade! The Internet is no longer a luxury. It is now a necessity. Every day, more and more commerce is conducted over the Internet, more businesses are built around the Internet, and more people use the Internet for their primary source of entertainment, communication, and social networking. To provide all this functionality, more and more web applications and services are available and required. These applications and services are replacing traditional desktop applications and legacy ways of doing things; the local computer focus is now Internet-centric. Sun Microsystems' motto, "The network is the computer," truly has become a reality.

The way today's web sites are developed and how the underlying architecture is implemented have also changed. With Web 2.0, web applications are much more dynamic than ever and offer rich, desktop-like functionality. Web applications that once ran exclusively on servers and produced HTML output for web browser clients are now multitiered, distributed applications that have both client components like AJAX (Asynchronous JavaScript and XML), JavaScript, and Flash, as well as server components like mod_perl, PHP, Rails, Java servlets, etc. These new web applications are much richer in features, and users now expect them to behave like desktop applications. The result is a satisfying and productive user experience.

The architecture that is required to support these applications has also changed. What used to be a simple database-to-web-application topography now comprises more layers and components. Functionalities that were formerly implemented in the web application code are now spread out among various services or servers, such as full-text search, caching, data collection, and storage. The concept of "scale-out versus scale-up" has become a given in web development and architecture. This is the case now more than ever before with *cloud computing*, which offers dynamically scalable services, either virtualized or real, over the Internet.

One component in all of these changes is caching. In terms of web applications, *caching* provides a means of storing data that would otherwise have to be retrieved from the database or repeatedly regenerated by the application server. Caching can significantly reduce the load on these back-end databases, allowing for better web application performance overall. Also, a database isn't the only

Chapter 1: LAMMP, Now with an Extra *M*

point of origin for information. Other sources of information could include remote service calls, search index results, and even files on disk — all of which can benefit greatly by caching.

Originally, there really was no easy way to provide good caching. There was a kind of caching using tricks like `IPC::Sharable`, global/package variables, database session tables, even simply files, but nothing offered real, centralized caching of the type that is available now.

This is where the extra *M* in this chapter's title comes in. It stands for memcached. memcached is a high-performance, distributed memory object caching system that provides caching for web applications. Along with covering the other letters of the LAMMP acronym — Linux, Apache, MySQL, and Perl — this book will also cover how you can leverage memcached in your web application development.

The object of this book is to show you everything you would need to know about MySQL, memcached, Perl, and Apache, as well as many other great technologies including Gearman, Sphinx, AJAX, and JavaScript, in order to take advantage of each for writing feature-rich, useful, and interesting web applications. This book also covers a lot of material that will expand your skill set to help you become a well-rounded web developer.

Linux

Linux is the world's most popular open-source operating system and the operating system on which a significant percentage of web servers run. Linux, originally created by Linus Torvalds starting in 1991, is itself a term given to the operating system, which includes numerous programs, utilities, and libraries around the core Linux kernel.

Linux was developed on and freely distributed over the Internet by a growing group of developers. It matured along with the Internet, emerging with the same principle of open development and communication that the Internet is known for. This open development concept, known as *open source*, or free software, is a model that allows developers to see the source code of a program and make modifications such as bug fixes and enhancements to the code. This model allowed for developers all over the world to contribute to Linux. This even included development to the kernel itself, as well as to the utilities and programs bundled along with the Linux kernel. Programs included compilers, interpreters, web servers, databases, desktop environments, mail servers, and many other tools that meant people could install an operating system that had everything they needed for implementing a web server, along with dynamic web applications.

Many programs that were available (and still are available) were made possible by the GNU Project. Initiated by Richard Stallman in 1984 along with the Free Software Foundation, GNU had the goal of creating a UNIX-like operating system with the philosophy that *"people should be free to use software in all the ways that are socially useful."* These tools, particularly the compiler GCC, were crucial to the development of Linux. Also crucial to Linux's adoption was the GPL (GNU Public License), which also came from the GNU Project. This license allowed developers to contribute to projects, knowing that their work would remain open and free to the benefit of the world.

Apache, Perl, PHP, and MySQL were developed to run on a number of operating systems. They also ran well on Linux, and with the same concept of open development, they allowed developer contributions to their advancement and maturation.

Originally, Linux was dismissed by many a pundit as being a "toy" operating system, or at best a "hobbyist" operating system. Nevertheless, system administrators, who quickly became Linux enthusiasts,

Chapter 1: LAMMP, Now with an Extra *M*

quietly deployed Linux to run an increasing number of services across the tech world. Ironically, many of the critical articles written by these skeptical pundits were probably being served at the time on web servers running Linux.

Today, Linux is considered a serious operating system. You can now buy hardware with Linux pre-installed from all major server vendors. Most interestingly, even big vendors who sell their own Unix variants also sell and support Linux on their servers — Sun, IBM, and HP are examples.

Without question, when a web server is installed and launched today, there isn't much thought as to whether Linux should be used — just as a desktop operating system is most of the time assumed to be Windows, a web server operating system can often now be assumed to be Linux. For several years now, even personal computers have been available with Linux preinstalled.

Although this book's target operating system is Linux (the *L* in LAMP), the author has attempted not to leave Windows Apache MySQL Perl (WAMP) developers out in the cold. Where possible, installation instructions and other configuration parameters are made available for Windows.

Apache

Another open-source project that had its genesis around the same time as Linux is the Apache HTTP Web Server. Developed by the Apache Software Foundation, the Apache HTTP Web Server is the world's most popular web server. Therefore it is also the web server that this book covers. Apache was originally released in 1994, around the same time that Linux was coming into popularity. Apache was most often bundled along with Linux in various Linux distributions, so setting up a Linux server usually meant you were also setting up Apache.

The pie chart in Figure 1-1 shows the market share of the Apache web server as used by the million busiest web sites, as of March 2009.

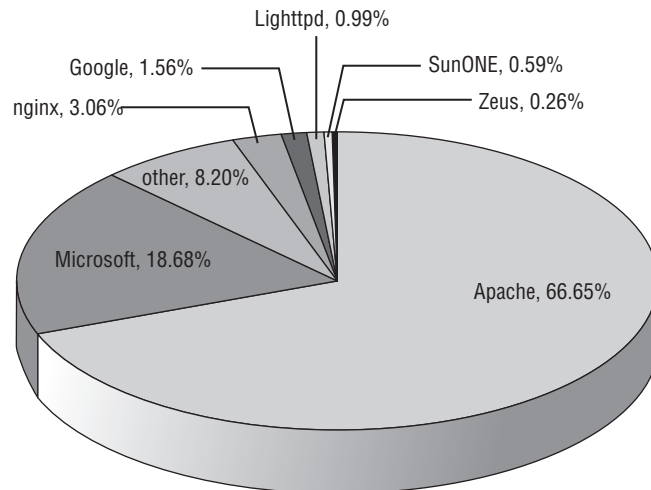


Figure 1-1
Netcraft, <http://news.netcraft.com/>

Chapter 1: LAMMP, Now with an Extra *M*

With a running Apache server, you had at your disposal a full-fledged web server that allowed you to build web sites — both static pages and dynamic web applications using CGI (Common Gateway Interface). Since then, Apache has evolved even further, becoming much more modular. The number of programming languages available for building web applications with Apache has also increased: You now have a choice of using CGI, `mod_perl`, PHP, Ruby, Python, C/C++, and others. For Java web application development, the Apache Software Foundation has developed Apache Tomcat, a JSP and Java servlet engine that can talk HTTP. So there are many choices for developing web sites, depending on what you prefer and where your expertise lies.

This book will focus on Apache web development using Perl, and in particular, `mod_perl`. Since Apache is very modular, it allows for developing various modules to extend its functionality, as well as providing access to the server to run various interpreted languages such as Perl, PHP, Python, ASP, and Ruby. This is in contrast to how CGI worked, which was running programs externally to the web server.

MySQL

Another of the open-source hatchlings is the MySQL database. MySQL was originally developed on Solaris but soon switched to be developed under Linux as Linux became more stable and more popular. MySQL grew, along with Linux, to become the default database of choice for web application development on Apache. This was because MySQL is fast, reliable, easy to install and administer. Also, it didn't cost a fortune (whether free or at the various support level pricings), and had various client application APIs and drivers, including Perl.

As far as web applications go, one change made during the last decade was MySQL's prevalence as the de facto database for open-source database development. Already quite popular a decade ago, MySQL has since advanced greatly in capacity, features, and stability to become the world's most popular open-source database. Most Linux distributions make it extremely easy to install MySQL (as well as PostgreSQL) during operating system installation, so you can have a fully functioning relational database system (RDBMS) that you can readily use for your web applications in no time.

Many popular web sites and customers use MySQL for a number of purposes. Figure 1-2, shows a list of the 20 most popular web sites that run MySQL.

Other sites and organizations that run MySQL include:

- Slashdot.org
- LiveJournal
- Craig's List
- Associated Press
- Digg.com
- NASA - JPL
- U.S. Census Bureau

This book shows you much more than previous web application development books. You will see just how powerful, yet how easy, it is to use MySQL. The author hopes this will give you a reason for making MySQL your database of choice, if it isn't already so. In this book, you will see:

Chapter 1: LAMMP, Now with an Extra *M*

- ❑ How to install and configure MySQL
- ❑ How to use MySQL's various utility and client programs
- ❑ How to use MySQL. This book starts out with simple usage examples for those who aren't familiar with databases and progresses to more advanced usage examples, showing you how to write useful triggers and stored procedures.
- ❑ How to use MySQL storage engines and what each engine is designed and best suited for
- ❑ How to set up dual-master replication — something you'll want to know if you are a web developer at a smaller start-up company. You can trust the author that this is a possibility in this industry!
- ❑ How to write a user defined function (UDF). Yes, this will be implemented in the C programming language, even though this book is targeted to Perl developers. Even if you are a true Perl geek, you'll probably find this interesting — possibly even enough to make you want to write your own. It's always good to expand your horizons a bit!

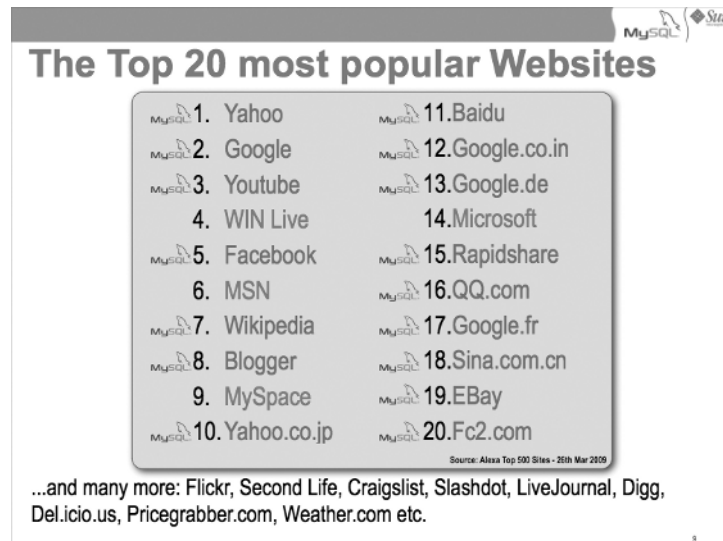


Figure 1-2
Sun Microsystems

memcached

memcached is a newer project, the new kid on the block, that came into being later than Linux, Apache, MySQL, or Perl. However, memcached has become just as much an integral component to the overall LAMP stack — which is the reason LAMP should now be referred to as LAMMP! Perhaps no one has thought of this yet because memcached is so simple to run and just works, or because it's so ubiquitously used that it almost goes without saying that it's now the de facto caching solution for horizontal web application development. That being said, memcached deserves some focus and appreciation for how it can benefit your web application platform, and likewise deserves a letter up on the LAMMP sign on the mountaintop above Hollywood.

Chapter 1: LAMMP, Now with an Extra *M*

memcached is a high-performance, distributed memory object caching system developed by Danga Interactive to reduce the database load for the extremely busy web site LiveJournal.com, which was at the time handling over 20 million dynamic page views per day for 1 million users. memcached solved for LiveJournal.com the problem that many other sites also have — how to reduce read access to the database.

A typical way to improve the throughput of a site is to store all query results from the database into memcached. Then, before fetching new data from the database, first check to see if it exists in memcached.

Using memcached, LiveJournal.com reduced their database load to literally nothing, allowing them to improve user experience. Because memcached was developed and released to the world as open-source software, Danga's creation has benefited thousands upon thousands of web developers, system administrators, and the wallets of numerous organizations due to hardware cost savings. Now it has become possible to utilize commodity hardware to act as simple memory servers. Some memcached success stories are discussed in the following sections.

Gear6

Gear6 is a company that built a business around scalable memcached solutions for superior site scaling, enabling their customers to scale their dynamic sites. Gear6 allowed these sites to increase their use of memcached (in some cases growing from about 100 gigabytes to 3 terabytes in only six months!) without using more rack space. memcached also helped Gear6 grow its customer base because of its wide use, as shown in the following table:

Type of Site	memcached Function
Social networking sites	To store profile information
Content aggregation sites	To store HTML page components
Ad placement networks	To manage server-side cookies
Location-based services	To update content based on customer location
Gaming sites	To store session information

Clickability

Clickability is a company that provides SaaS (Software-as-a-Service) web content management platform products. Their services include content management, web site publishing and delivery, search, web analytics, and newsletter delivery. They use memcached as a layer-2 cache for application servers to store content objects as serialized Java objects. They now run multiple instances of memcached, which are regularly cleared and versioned for cache consistency. They also use multicast messaging to cache objects across multiple memcached servers, as well as a messaging queue used for sending a clearing message to application servers. They originally did not use memcached, but were able to implement it into their architecture within a couple of days after deciding to take advantage of memcached's benefits. Because of memcached, particularly how it provides a caching layer to web applications to prevent excessive hits to the database, they now serve 400 million page-views a month!

Chapter 1: LAMMP, Now with an Extra *M*

GaiaOnline

GaiaOnline is the leading online hangout web site (with seven million visitors per month and a billion posts), geared toward young people for making friends, playing games, trading virtual goods, watching movies and interacting in an online community. A user can also create a virtual personality, referred to as an *avatar*. memcached has been a crucial tool in allowing GaiaOnline to grow their site from serving originally 15,000 to 20,000 users at a time to now being able to serve 100,000 users simultaneously.

How memcached Can Work for You

Gear6, Clickability, and GaiaOnline aren't the only memcached success stories. Some other sites that also use memcached extensively include: LiveJournal, Slashdot, Craigslist, Facebook, Wikipedia, Fotolog, Flickr, and numerous others.

In fact, Figure 1-3 shows that 80 percent of the top sites use memcached.

- LiveJournal
 - 20M dynamic page views/day
- Facebook
 - 80S memcached
- Fotolog
 - 40 memcached vs. 140 DS Serv. and 70 Web Serv.
- Flickr
 - 14 memcached vs. 144 DS Serv. and 244 Web Serv.
- Wikipedia
 - 79 memcached vs. 30 DS Serv.

1. Yahoo	11. Orkut
2. Google	12. Rapidshare
3. Youtube	13. Baidu
4. Live	14. Microsoft
5. MSN	15. Google.in
6. MySpace	16. Google.de
7. Wikipedia	17. QQ.com
8. Facebook	18. eBay
9. Blogger	19. Hi5
10. Yahoo.co.jp	20. Google.fr

Source: Alexa Top Sites - 08.05.16

80% of these web sites use Memcached!!!

Figure 1-3
Sun Microsystems

Indeed, memcached is a now primary component to the LAMMP stack. This book will attempt to show you why. Things you will learn in this book include:

- How memcached works
- What read-through and write-through caches are and can do
- Caching issues you should be aware of
- How to set up and configure memcached
- How to write Perl programs that use memcached
- The new libmemcached client library, which gives you even more performance for writing Perl programs that use memcached
- The Memcached Functions for MySQL, which are user-defined functions (UDFs) written by the author. These functions allow you to interface with memcached from within MySQL. You will see how you can use these convenient functions with MySQL:
 - From within your Perl code
 - With triggers

Chapter 1: LAMMP, Now with an Extra *M*

- ❑ With handy SQL queries that perform a simple read-through cache
- ❑ How you can modify your Perl applications to use these functions instead of using the Perl client to memcached
- ❑ Some simple caching strategies with memcached

Perl

The Perl programming language is the eldest of all the open-source siblings in the LAMMP stack. Created by Larry Wall — a linguist, musician, programmer, and all-around nice guy — in 1987, Perl was first developed for report processing and text manipulation. With the advent of the World Wide Web, Perl became a natural choice for developing web applications because of its innate ability to process and parse data. Implementing the functional equivalent of regular expressions or other Perl string manipulations, which are easy using Perl, takes many more lines of code and longer development time if implemented in other programming languages. This, as well as not having to worry about things like memory management, means relatively rapid development in Perl. You could write a fully functional Perl web application in a fraction of the time it would have taken to implement the equivalent application in the other programming languages available at the beginning of the World Wide Web. This is one of the many reasons Perl became popular for web development.

Originally, Perl web applications were written as CGI programs, which meant Perl programs were run by an external Perl interpreter. Drawbacks to this included a lack of persistence with running web applications; and running external programs could also adversely affect performance.

Then, in 1996, Gisle Aas developed and released the first version of `mod_perl`, which is a Perl interpreter embedded into Apache. Doug MacEachern, Andreas Koenig, and many contributors soon took the lead in developing `mod_perl` and released subsequent versions, such as version 1.0.

`mod_perl` now made it possible for Perl web applications to have persistence that was previously unavailable using CGI. Additionally, `mod_perl` gave Perl developers the ability to write Apache modules in Perl, because `mod_perl` is much more than CGI with persistence — it provides the Perl developer access to the entire Apache life cycle, including all phases of the HTTP request cycle.

A decade later we find that `mod_perl` is still being used extensively. The buzz and excitement may be over several new web development technologies and languages — and some would say Perl web development is passé — however, Perl is a more mature technology and it just works well — as is usually case with something that's been around a while. People are always excited about newer things, but there's still a lot to be excited about when you use Perl for web applications and development!

`mod_perl` 2.0, released in May 2005, provided many new and exciting changes, including support for threads, integration into Apache 2.0 (which itself had attractive new features and enhancements), the same great ability to write `mod_perl` handlers for any part of the Apache life cycle, and the added feature of writing `mod_perl` filter handlers for Apache 2.0's filter interface.

Certainly, other languages and web application development paradigms have some features over `mod_perl`. PHP has an application deployment model that has facilitated a bonanza of PHP web applications, such as Wordpress, Drupal, Joomla, Mediawiki, and many others, and particularly those with the APS (Application Packaging Standard) used in applications such as Plesk for web site hosting

Chapter 1: LAMMP, Now with an Extra *M*

services. This makes PHP application installation and deployment even simpler. Why has Perl/mod_perl not developed an equivalent of this? Perhaps it is because mod_perl already does give you as much control over the Apache life cycle and because it has a higher level of complexity (it's not solely focused on the HTTP response phase).

Also, you do have to have some ability to modify the Apache configuration if you use mod_perl handlers as your method of web application development. The answer is to use ModPerl::Registry, with which you can run CGI programs in mod_perl with very little modification to the application and still have all the benefits that a mod_perl handler has. Configuring Apache to run ModPerl::Registry is no more difficult for a web site administrator than loading mod_php to run PHP applications. So, where are all the applications? Well, we, as Perl web application developers, need to write them.

Here are some other reasons you might want to develop web and other applications using Perl:

- ❑ **Code is fun to write and free-flowing.** You can solve any number of problems in infinite ways while focusing on application development and implementation (the problem you're trying to solve) rather than on the language itself.
- ❑ **The Perl data structures work.** Both hashes and arrays are very easy when you go to organize data, navigate, and iterate. Try the equivalent in C, and you will see!
- ❑ **CPAN (Comprehensive Perl Archive Network).** You have a choice of modules for anything you could ever possibly want. So much functionality already exists that you don't have to reinvent the wheel. Every other day, the author finds an existing module that already does something he spent hours implementing!
- ❑ **Perl is a dynamically typed language.** For those who don't like to feel constrained, it's perfect. You can just write your program without referencing a document or web site to know how objects interface. Just code it!
- ❑ **Perl supports object-oriented programming.**
- ❑ **Perl clients exist for just about any type of server.** To name a few: MySQL, memcached, Apache, Sphinx, Gearman, and numerous others.
- ❑ **Perl has an XS (eXternal Subroutine) interface.** This allows you to write glue code and to use C code, if you need something to run faster than it would if it were written purely in Perl. This is what the MySQL Perl driver DBD::mysql uses for working with MySQL's client library.
- ❑ **Perl supports all the new exciting technologies, such as AJAX.**
- ❑ **There are numerous templating options.** You have various ways to tackle the site content versus application functionality.
- ❑ **You can even write Perl stored procedures for MySQL.** You do this using external language stored procedures, developed by Antony Curtis and Eric Herman.

Now, one claim you may have heard needs to be addressed: "Perl is great for prototyping, but you should develop the implementation in another 'real' language." This is a nonsensical statement that enthusiasts of other languages, having no experience in Perl development, have often said. Millions of dollars have been wasted completely reimplementing a perfectly good Perl web application to run in another language. Consider that many extremely busy web sites are running in Perl — Slashdot and LiveJournal are two such sites. The irony is that you will often see similar untrue statements ignorantly posted on the Slashdot forum — a forum that Perl provides so that opinions can be heard!

Chapter 1: LAMMP, Now with an Extra *M*

This book shows you numerous things you can do in Perl, including:

- ❑ A Perl primer for those of you who might be rusty
- ❑ A Perl object-oriented programming refresher
- ❑ Not just Perl web applications, but also writing utilities and command line programs
- ❑ Useful snippets of code that you can integrate into your Perl lexicon

You will also see how easy it is to use Perl to work with the other components of the LAMMP stack, for example:

- ❑ MySQL and memcached for data storage
- ❑ Apache mod_perl handlers
- ❑ Sphinx for full-text search (including the implementation of a simple search engine application)
- ❑ Gearman, which allows you to farm out work to other machines

It's the author's hope that this book will reinvigorate your fondness for Perl, or give you even more justification and enthusiasm for wanting to develop web and other applications using Perl.

Other Technologies

This book will also introduce you to other new technologies, namely Sphinx and Gearman. It will show you how to use these as additional components in the LAMMP stack to build truly useful and interesting applications.

Sphinx

Sphinx is a full-text search engine developed by Andrew Aksyonoff in 2001. It is an acronym for *SQL Phrase Index*. It is a standalone search engine, although it integrates nicely with MySQL and other databases for fetching the data that it then indexes. Sphinx is intended to provide fast, efficient, and relevant search functions to other applications. It even has a storage engine for MySQL so that you can utilize MySQL alone to perform all your searches. Sphinx also has various client libraries for numerous languages, including a Perl client library written by Jon Schutz, `Sphinx::Search`.

Sphinx also allows you to have multiple Sphinx search engines to provide distributed indexing functionality. This is where you would have an index defined that actually comprises a number of indexes running on other servers.

This book will not only introduce you to Sphinx, it will also show you a simple search engine application implemented using Sphinx, as well as a basic Sphinx configuration with a delta index that you could use for any number of applications that require a full-text search engine. You will also be shown how you can replace MySQL's full-text search with Sphinx for a better full-text searching functionality.

Chapter 1: LAMMP, Now with an Extra *M*

Gearman

Gearman is a project originally created (in Perl) by Brad Fitzpatrick of Danga, who is also known for creating both memcached and the social web site LiveJournal. Gearman is a system that provides a job server that assigns jobs requested by clients to various named worker processes. A *worker process* is basically a program that runs as a client and awaits an assignment from the Gearman job server, which it then performs. You split up your processing over various machines tasked for whatever requirements your applications need. This spreads out functionality, which is implemented in programs known as workers that might otherwise have been implemented in application code. This can also be used for MapReduce: distributing the processing of large data sets across numerous machines (for a great description of the MapReduce framework, see <http://labs.google.com/papers/mapreduce.html>).

This new functionality means web application developers and system architects can completely rethink how things have traditionally been done, using commodity machines to run some of these tasks.

Eric Day recently rewrote the Gearman job server, referred to as *gearmand*, in C for performance reasons, along with client and worker libraries in C. He has also written a package of new Gearman MySQL user defined functions based on the C library, and is working other developers for new and improved language interfaces. Another feature being developed is persistence and replication for jobs, which is one of the main things people ask about when first looking at Gearman for reliable job delivery.

This book will cover these new projects and you will see how to use them to implement automated data retrieval and storage, as well as Sphinx indexing through Gearman workers. This book also gives you one idea of how you can use Gearman to pique interest in Gearman.

The New Picture

Yes, things have changed in the last decade. And they probably will change more in the future.

Figure 1-4 represents how it is architecturally possible to implement the various tools and technologies that are discussed in this book. The architecture includes:

- ❑ memcached and MySQL, where a web application would retrieve its data: either durable data not cached from MySQL, or anything that needs to be cached within memcached.
- ❑ memcached objects, which are kept up to date to represent the state of the durable data in MySQL. This is done either by the application code or from within MySQL using the Memcached Functions for MySQL (UDFs), which would provide read-through and/or write-through caching.
- ❑ Sphinx, which can be run on a number of servers, provides the full-text indexing to the web application using the Sphinx::Search Sphinx Perl client module or through MySQL using the Sphinx storage engine. Sphinx has as its data source a query that returns a result set from MySQL that it in turn uses to create its full-text indexes.
- ❑ Gearman, which in this case is shown running on two different Gearman job servers (although it can run on any number of servers). Gearman is a job server for the Gearman clients — either

Chapter 1: LAMMP, Now with an Extra *M*

clients implemented within the application code, cron jobs, or clients in the form of the Gearman MySQL UDFs — to assign jobs to the Gearman workers. In turn, the workers can perform any number of tasks on all the other components, such as storing and retrieving data to and from memcached to MySQL, indexing Sphinx, or any other functional requirement for the web applications.

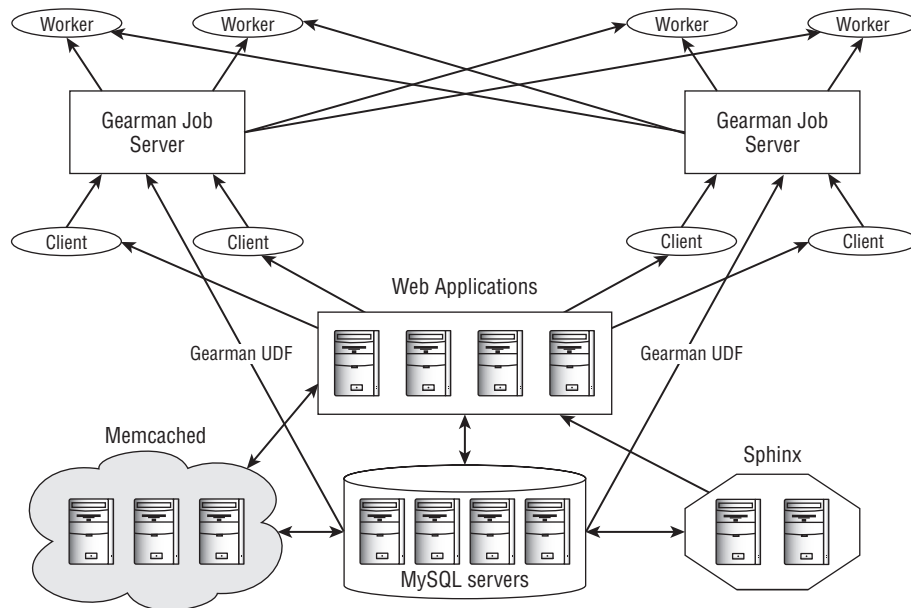


Figure 1-4

Variations on the theme that Figure 1-4 shows are infinite and limited only by your imagination. And this book hopes to provide some fodder for your imagination in this regard! Depending on your application or architecture requirements, your own version of Figure 1-4 will differ.

The Future of Open-Source Web Development and Databases

What does the next ten years hold for web development and the Internet in general? What features will MySQL, Perl, memcached, and Apache have implemented by then? Some things now are showing trends that are sure to continue:

- ❑ Open source is a proven development model and will continue to be the one of the major sources of innovation of new technology.
- ❑ MySQL has proven itself as a great back-end database for web applications and will continue to increase its market share, particularly because of its power, ease of use, and low or free cost, especially important given current economic conditions.
- ❑ Web applications will continue to evolve, developing more in number and variety of features. People will use many of these new applications in place of desktop applications.

Chapter 1: LAMMP, Now with an Extra *M*

- ❑ Cloud computing will increasingly become a preferred method on which businesses develop and deploy their web applications. This will depend on economic conditions, which may cause businesses to seek ways of cutting costs — hardware and hosting service costs traditionally being one of the largest expenses.
- ❑ SaaS (Software-as-a-Service), a new way of deploying software to customers as an on-demand service, will continue to grow. SaaS goes hand in hand with cloud computing.
- ❑ *Multitenancy* — users using the database at the same time — will work better and there may be development in this as a shared environment.

Projects to Watch!

The following are particular projects worth mentioning. These are projects that you will want to keep an eye on!

- ❑ **Drizzle:** Drizzle is a fork of MySQL version 6.0 that has the goal to become “A Light-weight SQL Database for Cloud and Web.” The idea of Drizzle is to create a very efficient, lightweight, modular database that is specifically targeted toward the Web and cloud computing. Many features of MySQL have been removed for efficiency’s sake, although some will eventually be reimplemented as long as their reintroduction doesn’t affect Drizzle’s goal of remaining lightweight and efficient.
- ❑ **MariaDB and Maria Storage Engine:** Maria is the next-generation storage engine based on MyISAM that provides transactional support, crash recovery, and the benefit of the speed for which MyISAM is known. MariaDB is a branch of the MySQL server that Monty Widenius and his team have released. It uses the Maria Storage Engine as the default storage engine. The goal of MariaDB is to keep up with MySQL development and maintain user compatibility, but also to keep improving the database and adding more features while engaging the open-source community in this effort.
- ❑ **Gearman:** With MapReduce becoming a household word, Gearman will increasingly play a significant role in distributed computing.
- ❑ **Apache Hadoop:** Similar to Gearman, this is a Java-based framework for distributed computing.
- ❑ **Perl:** Perl 6 will be released!
- ❑ **Percona:** Watch out for the great efforts of Percona. They are focused on providing their own high-performance branch of MySQL.
- ❑ **Hypertable:** A high-performance distributed data storage system, modeled after Google’s BigTable project.

Summary

This chapter introduced you to the topics and recent technological developments that this book will cover and it offered some observations about how much things have changed within the last decade. The suggestion was made that the LAMP stack needs to have an extra *M* added to it (to become LAMMP) because memcached has both benefited horizontal web application development and become a major component for so many web application deployments throughout the Internet — it is just as important a component as Linux, Apache, MySQL, and Perl. Also, this chapter offered some thoughts on what the next ten years may hold for open-source databases and web application development.

The author hopes you have fun reading this book. He had fun writing it.

